ADDITIONAL FEE:

Please charge any insufficiency of fee, or credit any excess, to Deposit Account No. 50-0427.

R E M A R K S

The Office Action issued November 9, 2004 has been received and its contents have been carefully considered.

The applicant wishes to thank the Examiner in charge of this application, Mr. Eric Coleman, for the courtesy and cooperation he extended applicant's undersigned counsel during the interview kindly granted on January 31, 2005. At this interview, applicant's counsel presented a proposed Amendment including applicant's extensive Remarks which are set forth, verbatim, hereinbelow. After discussing these Remarks and the cited prior art -- particularly, the U.S. patents to Morton and Potash -- the Examiner and applicant's counsel discussed a number of proposed amendments to claim 1 which would more clearly distinguish over this prior art.

Among these amendments was the following language proposed by the Examiner:

"wherein inputs and outputs of the arithmetic logic units (ALUs) are connected to the grid at the same time for receipt, processing and transport of the data in one clock cycle."

Claim 1 has now been amended to incorporate the substance of this feature. In particular, the microprocessor is recited as having "a clock" and "at least one arithmetic logic unit (ALU) having inputs and outputs." Furthermore, the circuit components are recited as being interconnected "on a grid of buses" and the inputs and outputs of the ALU are recited as being "each connected to a separate bus of said grid".

Finally, the ALU is recited as being "operative to receive, process and output data during one microprocessor clock cycle".

These limitations now clearly demarcate applicant's invention with respect to Morton and Potash. The operation of the microprocessor systems of Morton and Potash are described in detail in the discussion that follows. For the operation of the present invention, the Examiner's attention is specifically directed to the color diagram on page 39 hereinbelow and the discussion immediately following.

Applicant has also added new <u>dependent</u> claims 6-9 to particularly recite a number of unique features which have been disclosed, but not claimed, heretofore. By use of an X-Y grid of buses, with switch nodes at the points of intersection, it is possible to connect any component to any other component in a multiplicity of ways. The claims are directed to this unique capability.

Support for these claims is provided by Fig. 1 of this application which illustrates the X-Y grid.

All of the previously pending claims of this application -- namely, claims 1 and 3-5 -- stand rejected as being unpatentable under 35 USC §103(a) over the U.S. Patent No. 6,088,783 to Morton in view of the U.S. Patent No. 4,760,518 to Potash. This rejection is respectfully traversed for the reasons given below:

The key difference between the applicant's processor and the one developed by Morton is the way he uses the data switches as the core data transferring grid to transfer data from any Processor Element (PE) to another PE. A Processor Element could be a register, ALU, memory, Cache, Input/Output port, video buffer, etc.

There are endless examples for the use of data switches to transfer memory; however, the concept of using them in a grid as applicant has developed is unique. The use of crossbar switches have been around since the 1900s where they were first used in switching for the telephone industry. The typical crossbar switch routes data from a horizontal port to a vertical port or from a vertical port to a horizontal port. It cannot route data from a horizontal port to a horizontal port or a vertical to a vertical, whereas the applicant's system can do all of the above in one clock cycle.

To help explain the differences between the applicant's invention and Morton, it is first necessary to explain how Morton transfers data. Looking at Morton's Figure 7 one can see a block diagram representation of a crossbar switch that uses Mux (multiplexers) which are described on column 21 line 40 onwards. Also looking at Figure 8 one can see the Parallel Arithmetic Unit which connects to the crossbar 109 (Fig. 1). Morton uses 4 mux's to select 16 bits of a 64 bit data word which are fed into a Parallel Arithmetic Unit 110 to 113. See Col 22 - Line 5 to 15). Col 22 - Line 29 to 26, which states:

> 'The Crossbar switch is implemented from eight, unidirectional multiplexers, 702 to 705 and 707 to 710. It is logically organized in two parts, a Data Cache read section and a Data Cache write section. Since the Data Cache cannot be read and written by the Vector

Processors at the same time, the Data Cache Write Bus, or port, 701 and the Data Cache Read Bus, or port, 706, can be connected to form the 64-bit bus 125 to/from the Data Cache.'

The Data Cache 108 can only send or receive data from a Parallel Arithmetic Unit (110 to 113); it cannot send to one of the Parallel Arithmetic Units and receive from another unit. The bus is set up as a 64 bit data word which can be either sent or received so that each 16 bit Vector Processor bus bits must either be read or written at the same time. Also note that the write bus 701 and 706 are connected together which then connects to the Data Cache 108. The following points are made to explain how the applicant's grid allows data to travel in more than one direction and in one clock cycle.

Figure 1 of the attached drawings is from Morton's Figures 7 and 8 which have been pasted together to show clearly how the data is transferred from the Data Cache 108 to the Crossbar switch 109 and through to the Parallel Arithmetic Units (110 to 113). The Parallel Arithmetic Unit in Figure 8 of the Morton patent has one input and one output. Therefore, to perform a calculation on two words of data, for example A + B of which A and B are stored on the

Data Cache, first the value A would have to be read from the Data Cache 108 and stored on 'A Latch' 802 in the first clock cycle or clock stage and then in the next clock cycle or clock stage the value from Data Cache B could be stored on 'B Latch' 803. Mux A & B 804,805 would then transfer this result to the ALU 806 which could output the result to the Data Cache which is outputted from ALU 806 via VP(n) Memory Bus B 816. In order to store the value from the data bus to Latch A 802 data must go through the A Mux 804 and pass through the ALU 806 and be outputted into the register bank 801 which then stores the result into A Latch, since the VP(n) Memory Bus A cannot make a direct connection to either A or B Latch 802,803. This would take several clock cycles or clock stages assuming that there was no overflow (e.g., greater than 16 bits) in which case further cycles may be required. The Morton system does allow for the above mentioned calculation on four different ALUS using 110 to 113 separate 16 bit data words at the same time but to complete these calculations it would take at least two or more clock cycles or clock stages.

Accordingly, some differences in the design between Morton's and applicant's systems are as follows:

- Morton has only one input/output to each Parallel Arithmetic Unit 110 to 113, whereas the applicant's system has two inputs and one output to each ALU
- With Morton, data from Data Cache can only be read or written at any one time, whereas the applicant's grid has a number of data paths
- The Morton system cannot read and write from the Data Cache at the same time. This does not allow it to connect two or more ALUs together without providing additional clock cycles or clock stages because the data must be stored on the latches and cache prior to processing.

Figure 1 of the attached drawings shows only one Parallel Arithmetic Unit 110; there would be three more connected to each mux. All four Parallel Arithmetic units can only read or write data at the same time as discussed, whereas the applicant's system can have one ALU reading and one writing data.

Conventional processors only allow data to travel one way like a road freeway, where data can travel in either direction but in only one direction at a time, while many new processors use multiple buses but are limited in the way they can connect. Some examples of these are Single Instruction Multiple Bus Architecture PC where multiple buses are used to allow multiple data to be calculated at the one time. The applicant's system has an advantage in the way in which it can share data between these buses and also share processing elements.

Part of the inspiration for the applicant's design was on a Central Business District road layout where you have roads running vertically and horizontally. Each road can only have one vehicle on it at a time. Vehicles can cross each other at intersection points, but vehicles can move up a vertical road and then down a horizontal road and back down a vertical road as long as no other vehicle is using that road at that time. While the vehicle passes around, any number of people are allowed to read the vehicle registration but they are not allowed to go onto the road.

Turning now to the Office Action, there appear to be some misunderstandings of the applicant's invention and how the 'grid' works. In order to highlight the difference in operation between the applicant's system and Morton, reference is made to the attached Figures 2 and 3. Figure 2 is a drawing which shows a basic design that was used to help explain how the architecture works. This design is Figure 1 in the present patent application but some identification numbers have been added to some of the buses to help explain how the data is transferred. In Figure 3 of the attached drawings the microprocessor 10 is shown with some of the intersection nodes activated to allow data to travel to the appropriate Processor Element (a processor element could be a register, memory, address register etc).

In Figure 3 the following operations occur.

(1) Internal Memory 1 36 reads memory from the address provided from Address Register 1 18

(2) Internal Memory N 38 reads memory from the address provided from Address Register 2 20

(3) Data from 'Internal Memory 1' 36 and 'Internal Memory N' 38 are inputted into ALU 1 30

(4) The result from ALU 1 30 is fed into Register 1 14 and also fed into External memory N 42

(5) Data from Internal Memory 1 36 is also fed into External Memory 1 40.

The number of clock cycles required to do this depends on how the ALUs 1-N (30 and 32) are constructed and how the microprocessor 10 is constructed. Some examples are illustrated below:

ALU 1 30 does not have any internal registers which store data at the two inputs (say inputs A & B) therefore the result of the two inputs from the Y-Bus lines 51,53 can be outputted to Y-Bus 50 in the one clock cycle; but before the device can store this value the ALU 1 30 would have to be allowed enough time to allow the result to be outputted. The issues of ALUs returning a result in the one clock cycle have been covered in the present specification on pages 7 to 10. Most ALUs perform calculations which are passed directly though logic gates therefore these calculations can be performed very quickly. Looking at the node 103 on the attached <u>Figure 3</u> one can see that the N size bus switch has closed which allows data to travel from the ALU 1 X-Bus 50 to Y-Bus Line 64. In this example the Instruction Set

Decoder 34 allows the data at Y-Bus 64 to be stored on register 1 14, but if the result was not required to be stored on the register then the Y-Bus 64 could still be used to transfer data without register 1 14. Y-Bus 64 transfers data to X-Bus 60 via node switch 114 so it can be seen that data from ALU 1 30 has done a U-turn from X-Bus 50 to Y-Bus 64 then to X-Bus 60. From the Examiner's remarks on Page 5, paragraph (c) of the Office Action, he appears unclear as to how data can travel from a component connected to a Y Bus and travel to another component of the Y-Bus. The above example clearly shows that it can. Also, since each node switch (eg Nodes 103 to 713) is only a bi-directional data switch with N switches per node then data can travel through many switches in the one clock cycle. The number of switches the data passes does not mean that any extra clock cycles will be required.

Node 702 and 710 which are connected to Y-Bus 76 and are used to transfer data from Internal Memory N 38 to ALU 1 30 Input B. Address Register 5 26 is not used in this example so we can use Y-Bus 76 to transfer data from one Internal Memory N 38 to ALU 1 Input B 30. The Instruction Set Decoder 34, which is described in the present specification, would notify Address Register 5 26 not to read or write any data so that the Internal Memory N 38 can use the bus. It is as if the Address Register is not there and the Y-Bus 76 can be used for other purposes. This shows the difference of the applicant's system, which allows any bus to be used to transfer data around from the system of Morton. Internal Memory 1 36 data is outputted onto Y-Bus 66 so that External Memory 1 40 can read data from Internal Memory 1 36 and all that is required to do this is for the bus switch at node 208 and 212 to close and the ISD 34 needs to send a Write command to External Memory 1 40.

Morton talks about components that are interconnected on a grid, whereby a plurality of components can be switched under program control to a predetermined selection. Any component can connect directly to another component, via a connection of a node as discussed in the applicant's specification and recited in applicant's claim 1. It is also important to understand that where it mentions a predetermined selection of interconnection, this predetermined selection can be defined by hardware or software as described in the present specification where it talks about loading of the instruction set that can define how the grid is connected. Morton's patent, as mentioned above, needs to pass data through a number of components to transfer data. Cross bar switches have been around since the 1900s, but the core of the applicant's system is the grid of nodes in a uniform fashion with a plurality of components around the grid. This provides for any two components to be connected directly and even several components to be connected directly without the need to pass through any other component besides the data switch. The applicant's system does not see the difference between address bus and data bus since a data bus can be used to transfer an address register data and an address bus can be used to transfer data bus information. A bus in the applicant's system can be used to transfer data; the attached component to the this bus does not need to use that data but the bus can be borrowed to allow for data transfer via the borrowed bus.

The patent to Potash discloses a computer that superposes vector and scalar operations. The main aspects Potash discussed below are the differences between the applicant's system and Potash's XBar 40 and Vector XBar 44 which is embodied as the buffer 28. Buffer 28 is shown in more detail in Figure 4 of Potash patent. This diagram is explained in detail under the buffer section of the patent (Col 9, Line 34 to Col 15, Line 2).

Attached is a copy of Potash Figure 4 with some labels to help gain an understanding of how data is transferred in Potash. This Figure 4 shows the functional bus connections Y0,X0,Y1 & X1 and also the Memory Data busses MSC, M1 & M0. Buffer 28 includes in it a number of registers, two ALUs and a number of multiplexers as well as tri-state buffers as described by Potash. Figure 4 shows BussesY0,Y1,M0,M1 and MSC Bus which are inputted into registers 80 to 84 and these buses are then outputted via tri-state buffers 98a,98b,98t,98g & 98h. In order to transfer data from the functional Units 18 to 26 (Fig 1. Potash) to the Interleaved Memory Units 8 to 16, data must pass through buffer 28. Potash Figure 1 is a simplified diagram showing the Scalar and Vector Xbar 40,42 which is used to transfer data. In order to understand how this varies from the applicant's design, it is necessary to examine the XBar construction which is explained in Figure 4. To transfer data from the functional Units 18 to 26 (Fig. 1 Potash) to the Interleaved Memory Units 8 to 1 would require a connection via the Functional buses (Y0 or Y1) to buffer 28 and out of buffer

28 into Memory Data Busses (M1, M0 or MSC). This is a basic transfer of data from one bus to another bus. It will first be explained how the Potash system performs this data transfer; thereafter, the differences with the applicant's system will be discussed. If one wanted to transfer data from Y0 to M0, the following steps are required:

(1) Register 81 needs to latch the data from Bus Y0 which would take one clock cycle or part of the first clock cycle. (Clock St 1)

(2) Then the corresponding Mux 85 to 87 would need to select the data from Register 81

(3) The result of this Mux would be stored in Vector Register 42 this would need to be latched onto the memory of the register and then outputted in a later part of the clock cycle or another clock cycle. (Clock St 2)

(4) The corresponding Mux 108 then selects the appropriate vector register from vector register 42. Again this would either be done in another clock cycle or a later stage of the previous clock cycle. (Clock St 3)

(5) Latch 95 stores the result from Mux 108. Again this would either be done in another clock cycle or a later stage of the previous clock cycle. (Clock St 4)

(6) The result of latch 95 is then selected via mux 103 and outputted via a tri-state buffer 98a.

(7) There would be additional clock cycles and clock stages required to read the data from the functional memory units 18 and also additional cycles to store the data in the interleaved memory units. We will not discuss these clock cycles since we are only looking at the difference of how Potash XBar works in comparison to the grid described in the Applicant application. We have described how data passes from one bus to another via a number of clock stages where as the Applicant system does not require any clock stages.

It is clear from this example that a number of different stages must be passed through to transfer data across Vector XBar 44. Without knowing the detailed timing diagrams on how the different stages latch data into the corresponding registers and latches, it is hard to say how many clock cycles would be required by Potash, but it is known that 4 or more stages may be required. Even if Potash uses one clock cycle then that clock cycle would need to be separated into 4 sub cycles which allow for the appropriate timing of the registers and latches to be stored. A register or a latch requires a signal to notify the register or latch to store the data on its inputs. The data needs to pass 4 latches/ registers the timing of this data transfer requires complex timing from the clock generator.

Looking at the attached Figure 5 we see a drawing of one embodiment of the applicant's specification which shows the transfer of data from Internal Memory N 38 to Internal Memory 1 36. In order for this to occur two data node switches need to be closed, that is node switch 708 and 710, this would allow data to travel from X Bus 56 to Y Bus 76 and then to X Bus 54. Since both nodes are switches then the data travels from Internal Memory N 38 to Internal Memory 1 36 almost instantly, there would be some delay as the data passes a node switch. This delay would be the equivalent time taken for data to pass though a tri-state buffer which would mean that data can travel thought the grid much faster than the Xbar used by Potash, since Potash uses a number of registers/latches and also Muxes to transfer data through the Xbar. The applicant's grid only has two data switches which close to transfer data through a grid and although we describe in the applicant's specification that one can design the grid with the use of multiplexers instead of bi-directional tri-state switches this design still does not require registers or latches in the grid to hold data unlike Potash. The applicant' system as described allows for the direct connection of two ALUs as described in the applicant's specification on page 9 line 23 onwards.

The applicant's system allows for any switch to be closed to allow data to transfer from an X bus to a Y Bus the switches as described in the applicant's specification could be N bi-directional tri-state switches for each node where a node is a intersection of an X and Y bus. The other way to provide switching would be to use multiplexers with tri-state outputs where each multiplexer would be N bits wide to provide for a N bit data bus. As shown on figure 6 in the applicant's specification we show one multiplexer which can select any register, in this example we only show

4 registers and no other components for simplicity and only show input A and do not show the other input B to the ALU 30 or the output of the ALU 30. As described in the application you can provide more multiplexers which operate in parallel to multiplexer 80 and provide the same connections but can select different registers or vertical buses. If we have two multiplexers connected in figure 6 and the first multiplexer is connected to register 2 16A and the second to register 3 16b one might say that this does not provide any advantage since you can only load one data word at a time to the input A of ALU 30. But the advantage here is that one multiplexer can read from register 2 16A and output the data to the X bus as shown on figure 6 once the data is on that bus ALU 30 can read that data if needed. The second multiplexer can then read the data and feed it to any of the other unused registers via a connection a Y-Bus. So you can read from register 2 and output data to ALU 1 and also to Register 3, if you add a third multiplexer then you can read one register and output data to two registers and also to ALU 1. The multiplexers described would need to be bi-directional tri-state multiplexers; otherwise if one were to use uni-directional tri-state multiplexers then one would need to

use two multiplexers for each node since one will need to allow for data to travel either direction. The preferred embodiment described in the present application use the N bi-directional tri-state switches for each node as described here. The engineer's choice to use either multiplexers or bi-directional switches depends upon how many node connections are required and how flexible a system should be. Although the best solution is to allow for data to travel in any direction by providing for an N bi-directional tri-state switch for each node so that the system can be developed and then the programmer can find the optimum connections for the application at hand.

The above explanation as to how the applicant can use either multiplexers as one option to transfer data and also the use of bi-directional tri-state switches to transfer data, is intended to show the differences between the applicant's system and Potash and Morton. Potash uses multiplexers in the middle of registers and ALUs its Xbar system where as the applicant's system does not.

Turning now to the Examiner's comments (a) to (h) on pages 3- of the Office Action, applicant responds respectfully as follows:

Point (a)

The Examiner contends that someone skilled in the art would be able to combine the teaching of Morton and Potash. Applicant does not believe this is possible since both the Morton Crossbar switch and Potash XBar operate very differently from applicant's data bus grid, as previously discussed. The main differences are that Morton only has one bus connection to the ALU (Parallel Arithmetic Unit 110 to 114 of Morton's patent) which means that, to add two data words together, one must send one word at a time on the bus and then, once the result has been calculated, one must read this data. This means that to add two data words together it would take at least 2 clock cycles or clock stages since the data bus cannot read and write at the same time. The Morton system provides for data transfer via a data bus which can only be read or written at any one time as shown on Morton's four 16 bit data bus 128 which make up for a 64 bit data word. This limits allowing parallel buses to read and write at the same time.

The Potash system is different to the applicant's system in the way it processes data as described above.

Potash needs to have the data pass through many stages of data transfer from one side to the other side of the Xbar. The Potash system, like Morton's, has ALUs which need to have both the A & B inputs of the ALU stored one at a time prior to any calculation. The result cannot be read until the next stage (e.g. clock cycle). The reason for this is that Morton and Potash both use latches or registers to store the A & B inputs for the ALU and it is not until the inputs have been stored that a result can be made. Also since the A & B inputs are made via the same input, one must store A and then B in the next clock cycle or stage. Applicant' system allows direct connection of both ALU inputs (A & B) to the necessary memory or register, so that one can return the result in the same clock cycle. (As with all ALUs one would have to wait for the time to make the necessary calculation.) The result of applicant's ALU can then be stored once the result has been made. It can be seen that the way in which applicant uses ALUs, and in the way applicant's system allows for the ALUs to make direct connections to multiple components, are very different from both Morton and Potash.

Point (b)

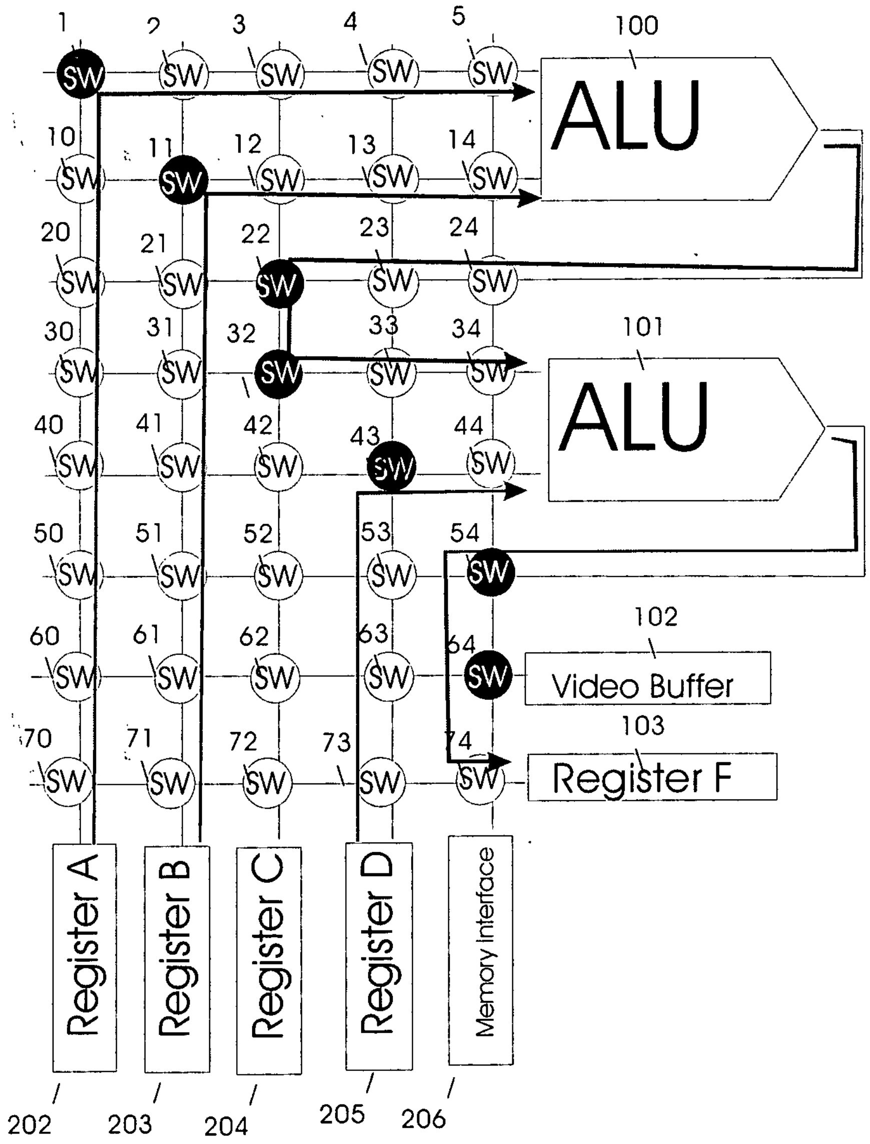
The Examiner comments on applicant's response to the previous Office Action and comments on the figures in this previous response. The Examiner contends that 'the art of comprising a crossbar switch was taught and Potash was well known in the art to comprise horizontal and vertical conductors that where selectively connected via a grid or matrix if connection points'. Applicant's grid and the Morton crossbar are very different since applicant does not rely on multiplexers to transfer data as shown in Figure 7 of Morton. Applicant's design uses N bi-directional tri-state switches for each node. Each intersecting Y-Bus and X-Bus has a switch which can transfer data from one bus to another and multiple switches can be activated to transfer the same data to several data buses, whereas Morton can only transfer data from to the horizontal to the vertical. As explained above, Morton cannot read and write at the same time since the outputs and inputs of the data cache are connected together, whereas applicant's system can have two parallel buses which can have data traveling in either direction. Morton's design only has one bus connection for each Arithmetic Logic Unit 110 to 113 so that in order to add two data words together from the data cache 108, Morton's design, as he describes it, must transfer one word at a time to any one Arithmetic Logic Unit. To store the result in the Data Cache 108 one would have to wait another clock cycle since the Bus 128 cannot be read and write at the same time. Therefore sending data to and from the Parallel Arithmetic Unit 110 to 113 is very different from applicant's design, since applicant's system can store the result of the ALU in the second half of the first clock cycle. Even if the ALU and the memory are both on the vertical buses, this does not limit them from transferring data between themselves, whereas a crossbar does limit vertical to vertical connections.

Point (c)

On page 5 of the Office Action, the Examiner mentions under Point c) "that (Since the figures in the instant application provide the ALUs connections parallel to each other either both horizontal or vertical) and memory at right angles then the connection would have provided connection of the ALUs to memory and not to each other)". The example given still embodies applicant's system, as claimed. Because there are so many combinations of the position of the ALUs, registers, etc., applicant has shown one basic design in the application; however, the body of text in the application and the claims cover a plurality of combinations. The following example was used to show how data travels in applicant's grid in comparison to Morton's and Potash's.

The present application does allow for vertical-to-vertical connections through the use of an intersecting horizontal connection. As will be illustrated below, even if both ALUs were on the vertical side then one could still cascade data from one ALU to the next ALU in the one clock cycle assuming that the clock cycle is long enough to allow the data to travel from one ALU to the next. (This is discussed in pages 8 to 10 of the application). Although the following diagram is not present in the application, it is included below to show how applicant's system can provide vertical to vertical connections.

Detailed examination of instant application grid



Data from Register A & B is fed into the ALU 100 and outputted into ALU 101 which adds to it Register D and the result is outputted to Register F.

Typical code:
Reg F = Reg A + Reg B + Reg C

The difference here is that applicant does not use a traditional crossbar switch which transfers data from the vertical side to the horizontal side or horizontal to vertical side. By the use of N switches which can individually be selected in any order, the grid can transfer data from the vertical side to the vertical side and the horizontal side to the horizontal side in any combination of the above. (Since all switches are bi-directional switches).

- Any horizontal component can be connected to any vertical component
- Any horizontal component can be connected to any horizontal component
- Any vertical component can be connected to any vertical component
- Multiple components can be connected together to share data in the one clock cycle
- Multiple combinations of the above can be performed in the one clock cycle

The Examiner does explain how, in order to connect ALUs to each other, more than one connection would be required.

The data can pass through all the connections in the one clock cycle so this does not affect the processor. The diagram above is only one of many possible designs. Although it shows only one combination, it is only an example of a plurality of components that can be connected on the vertical and horizontal axis. (See Page 12, line 10 of the present application).

Point (d)

The Examiner states, "Morton and Potash provide a means to directly connect the attached devices that crossbar of Morton and Potash provides a means to directly connect the attached devices whether the devices are ALU or memory or something else. By merely locating the connected device a connection at right angles to themselves then the transfer is readily performed in a single cycle".

The Potash system has embedded ALUs into the buffer 28 and the ALUs are embedded into XBar. This can be seen by looking at figure 4 of the attached diagrams which shows the Potash construction of the interconnection of the XBar and ALUs. As may be seen, one cannot move the ALUs at right angles since the two ALUs are configured in a way to perform

Scalar and Vector additions which was the purpose of the Potash invention. The invention does not provide connections at right angles since data comes in from the Y0,Y1 and MSC bus and from this bus it needs to pass a number of Multiplexers, registers, latches prior to being able to be connected to an ALU before it is outputted on Y0,Y1 and MSC Bus. The Examiner notes here that, just because the data passes through a number of components as mentioned, the invention still has the capacity to perform the operation in 1 clock cycle. Although this is true, it is important to understand, that in order for Potash to perform this operation, a number of sub-cycles or sub-clock stages would be required and this would require complex timing of the XBar and associated components. In contrast, the applicant's system can perform ALU calculations in 1 clock cycle and it can provide the two A & B inputs immediately without any sub-clock stages since both A & B inputs can make direct connection to any component connected horizontally or vertically via a number of node switches. The Potash system would take a lot longer, as will be explained below.

Referring to attached Figure 4, which is Figure 4 from the Potash patent, we will consider adding two data words from two of the three data buses provided Y0,Y1 & MSC (Note bus M0 & M1 cannot be connected to either ALU).

To perform a calculation using ALU 128

- First, the first word of data must be read from Y0,Y1 or MSC, selected via MX 89 and stored into Address register 34 which then needs to output the data to latch 124 so that it can be stored ready for ALU 128.
- The second word of data must be read from Y0,Y1 or MSC and selected via MX 89 and stored into Address register 34 and then outputted to latch 131. Only then can the ALU calculate the result.

To add using ALU 124 the same system of data flow must occur but with different components.

This shows that data cannot pass through a direct connection from the bus to ALU, whereas applicant's system can provide a direct connection to the ALU, which cuts down the number times required to perform a calculation. It needs to be understood that, since applicant's system removes the requirement to have these complex stages prior to a calculation, both applicant's system and Potash cannot be configured to operate as each other by merely moving around ALUs. The Potash system was designed to perform a task of adding Scalars and Vectors together, whereas applicant's system was designed to be a standard microprocessor that can be configured to suit the application.

This difference will be explained by way of an example. Assume one has a box of ALUs, Multiplexers, registers and latches, as well as N size bi-directional data bus switches which are all the same. These various components may be used to create the Potash buffer 28 which comprises the XBars and ALUs. One can also arrange the components in a configuration as per Figure 2 of the attached documents, which is the same as in the present patent application. Now assuming that both processors ALUs, multiplexers, registers, etc., operate the same way, then one can see that the Potash system requires several more sub-clock stages than applicant's system. The Potash system would require a longer clock cycle to allow for this to occur, whereas applicant's system can direct data from any memory or register immediately since the bi-directional data switches have such a small delay, as compared to multiplexers, registers and latches. Although both processors can perform calculations on two data words, it is the way in which this occurs that is very different. All processors can perform ALU calculations, but there are many ways in which data from the bus is transferred to the ALU. Applicant's system does not rely on any one switch to transfer data since any number of switches can be used to route data whereas the Potash system requires data to be transferred through a number of multiplexers, latches and registers as described. It can be seen from the attached Figures 7 and 8, which show an example of two data words being inputted into the ALU and then outputted to the data bus. Figure 7 shows the Potash clock sub cycles required to perform this operation where clock cycle A has a number of sub clock cycles B to M. There are 12 sub cycles which are required to occur in one clock cycle, whereas applicant's design, as shown in Figure 8, occurs much faster assuming that the speed of the components used are the same in both examples. Figure 8 shows a possible clock cycle operation for applicant's design as shown in the attached Figure 6. In the first part of our clock stage 0 you can see that node switches 702,710,603,605 and 304 are closed to allow data to flow as illustrated in Figure 6 and this can occur very quickly. Once these switches are closed they can remain closed for the duration of the clock cycle N. Clock stage P is the time required for the ALU to perform the calculation, which is the same as clock stage H in the Potash timing diagram. Thus, one can clearly see the difference between the two operations are all the other operations which are required to be performed in Potash's system which is unlike our system.

Morton's system, like that of Potash, only has one input from the data bus to the ALU 110 to 113 (Arithmetic Unit) so this means that to perform a calculation on two data words one must send each word one at a time rather than send both to the ALU at the same time as in applicant's system. Morton's system was designed for a different application than the present system and Potash, in that it was designed as a DSP chip. As previously described, Morton's system does not allow for the ALU 110 to 113 to have data inputted into and outputted immediately since there is only one bus in and out of each ALU 110 to 113. This requires sending the two input words one at a time and then, after that, the bus must not be used so that the ALU can be read. As with Potash, in order to do this it would

Morton's system, like that of Potash requires several sub cycles similar to the attached Figure 7.

The diagram above shows the same information as the previous diagram but each data path is shown in a different color to illustrate how the switches transfer data. It is important to see how data passes through the whole data bus. If one looks at the green data path, one can see that the output of ALU 100 passes through switches 20,21,22,23,24 but only switch 22 is closed which diverts data from switch 22 to switches 3,12,32,42,52,62,72 but only switch 22 and 32 are closed so data then travels towards ALU 101. Looking at Register C, if one wanted to store the result of ALU 100 then one could store it at register C 204 since that bus holds the result of register C 204. If one wanted to store the result at register C 204 then a corresponding write command would be needed to send it to register C 204. But in this example, register C 204 does not write data. Any component can connect to the data bus to read data but no two components are allowed to output data onto the same bus as this would create a data bus crash. This is an example of the present invention as defined in the claims of this application. By showing different colors this figure shows how a node switch transfers data onto a X or Y bus which can then output the date to the component attached the bus or divert it through another switch.

Point (e)

The Examiner contends that the figures shown in applicant's previous response to the previous Office Action do not correspond to the drawings provided in the original application, since the drawings in this application show only one configuration of ALUs, Registers, memory etc. The attached diagrams illustrate how the grid connects the components together, as compared to where Morton's Crossbar and the Potash Xbar which operate differently as described.

The previous description of how one can have redundant switches was included to show how the flexibility of the presesent invention allows for redundancy. Looking at the original application Figure 1, a programmer could program checks into the software to check nodes against each other so that, if a faulty node were detected, then the software could re-write the instructions to allow the data to bypass the fault. In a conventional PC which allows you to scan the hard drive for errors, although this may not have been the original intention of the PC when it was designed, an experienced programmer saw the opportunity in the system to allow for software management for a hard drive error. The PC scans each sector of a hard drive testing it for errors; when it finds an error it does not allow the operating system to store data in that location and thus a software solution is made to a hardware fault. The same can apply to the present design and although it was not described and directly claimed. An experienced person in the art of programming would allow for such error checking if desired in the present system. The grid of the present invention allows for a number of different ways in which data can travel, thus allowing for redundancy.

Point (f)

In this point the Examiner makes reference to applicant's previous response that Morton and Potash taught a system that provides for connection between devices for sharing data and, Potash taught that the first Xbar could be connected to the second Xbar. The Potash system has two Xbars, one for Scalar data and one for Vector Data; thus there is a Vector XBar and a Scalar XBar. The Potash system must cascade data through a number of components as shown in Potash's Figure 4, where these components comprise of latches, registers and multipliers and data must pass through a number of these components in order to provide for the data transfer from one side to the other side of the XBar. Figure 1 of Potash shows the Vector XBar 40 and the scalar XBar 44, but one must to look at Figure 4 of Potash to understand how the XBars work. When looking at this figure we see that the design it totally different from the grid of the present invention.

Point (g)

The Examiner notes that, in applicant's previous response, applicant commented that the Potash system must pass data through the XBar which has temporary memory for storing intermediate results, whereas the inventive system does not require an intermediate stage as allegedly shown in Figure 1 of Potash, and the inventive system can directly transfer data directly from memory to an ALU and can output the result all in one clock cycle as allegedly shown in Figure 4 of the application. There needs to be a clear understanding of how Potash transfers data, as Figure 1 leads one to believe that data can travel from one of the buses directly to the ALU 30. Before one makes this assumption, one must understand how the XBar works and the ALUs work, which are all inside the buffer 28 Potash describes. To do this one must look at Figure 4 and read the corresponding Potash text to see how data travels; from this it can be seen that data must pass through a number of stages prior, to being outputted, as applicant described above. In the attached copy of Potash Figure 4, which is labeled <u>Figure 4</u>, it is seen how the ALUs 128, 129 connect back to the input of Muxes 88, 89. This shows that the output of the ALUs must connect back to the input Mux 88, 89 which means that <u>the output of the data must past through a number of components before it can be outputted on either X0 or X1 bus</u>. Figure 4 also shows that there is no direct connection from one ALU to another ALU, whereas the applicant's system can have direct connection via a number of node switches, and these node switches do not need to latch data or store data on registers prior to any calculations (unlike the Potash system which required latches 124,131 & 123, 132 to store data prior to outputting). Also the scalar register and address register are required to store the data prior to outputting and, since both these registers only have one input, then only one word can be stored at any one time. Therefore either two clock cycles or two sub cycles of a clock would be required, because they must be stored on the registers prior to being read. The term "sub cycle" should be understood to mean that the system has a main clock cycle and a number of clock cycles inside the main clock cycle. Therefore, the more sub clock cycles there are, the more complex the clock generator would have to be and a longer time would have to be taken to perform such calculations.

Point (h)

In the previous Office Action, the Examiner contended that it would have been obvious to combine the teachings of Morton and Potash to provide for a computer system with a plurality of node switches to interconnect the components. It is very difficult to see how one could, from both Morton and Potash, develop a grid of X and Y Buses with data switches at the intersecting points when these references do not use bi-directional tri-state switches but use multipliers with latches and registers. Potash does not have an X by Y grid, and even if Potash added a plurality of XBars using the configuration shown in Figure 4 of his patent, this would still function as the Potash system but different from the system according to the invention.

The attached Figure 6 gives an example as described in the present application, showing the data paths used to make the ALU to ALU connection that is shown in Figure 4. The following function is performed in this example:

Table 1:

ALU 1 30 = Reg N 16 + Internal Memory N 38

ALU N 32 = ALU 1 30 + Address Register 1 18

Register 1 14 = ALU N 32 Output

As shown in this table, the output of Register N 16 is added to Internal Memory N 38 and the result of that added to address register 18 and outputted to Register 1 14. Neither Morton nor Potash allow for direct connection of ALUs, in order to perform adding of three numbers they would require the use of the ALUs and latches or registers to hold data before it is outputted. Since Potash ALUs are embedded in the XBar, data would take a longer sequence of events to be performed. The system according to the invention has a direct connection from the Register N 16 and Internal Memory N 38 to ALU 1 30. Although input B of ALU 1 30 uses two node switches, this can be done since these switches do not store or latch data. Data passes though them as though the switch is transparent. This also gives a good example of how horizontal to horizontal data can be transferred, whereas Potash and Morton do not allow for such transfer. Figure 6 shows how bus 76 connected to Address register 5 26 can be used to transfer data while that register is not being used; however, if it was in use, then another vertical bus could be used to transfer data from the horizontal bus 56 to 53.

If one looks at address register 2 20 one can see that data from that register is fed down bus 70 and the connection node switch 409 allows data to travel to internal memory N 38 via bus 82. This provides the address for Internal Memory N 38 to read, which is outputted to ALU 1 30. When bus 70 is not used, then any other bus can connect to it if needed to transfer data around the processor.

Morton's Parallel Arithmetic Units 110 to 113 only have one bus connected to each unit so data can only be read or written at any one time. Therefore this does not allow the

ALU to feed two data words to them at the same time. Also Morton's data cache does not allow data to be read and written at the same time as discussed previously.

Accordingly, it is believed that claim 1, the only independent claim in this application, distinguishes patentably over Morton and Potash, taken individually or in combination. This application is therefore believed to be in condition for immediate allowance.

Respectfully submitted,

By [illegible]
Karl F. Milde, Jr.
Reg. No. 24,822

MILDE & HOFFBERG, LLP
10 Bank Street - Suite 460
White Plains, NY 10606

914-949-3100

I hereby certify that this correspondence is being deposited with the United States Postal Services as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450
on MARCH 2, 2005
By [illegible]
Date MARCH 2, 2005